Documentazione di Progettazione

Documento contenente la progettazione del sistema, includendo i diagrammi delle classi e di sequenza per le principali interazioni.

0612707416 - Gioia Iannuzzi
0612708273 - Luigi Montonetti
0612707862 - Arianna Paletta
0612708294 - Debora Villano

Anno Accademico 2024/2025

Indice

1 Introduzione

Il proposito di questo documento è descrivere la progettazione dettagliata di un sistema software realizzato per gestire feature correllate alla gestione di una rubrica di contatti, come la creazione, modifica, eliminazione e salvataggio dei contatti, con particolare attenzione all'adozione di pattern architetturali consolidati e best practice di ingegneria del software. Il sistema, oggetto di analisi, è basato sull'architettura **MVC** (Model-View-Controller), che separa chiaramente le responsabilità tra i dati, l'interfaccia utente e la logica di gestione degli eventi. Nella progettazione di queste funzionalità è stato utilizzato un approccio orientato agli oggetti, sfruttando strumenti di modellazione come diagrammi UML e interfacce grafiche sviluppate con FXML. Ogni interfaccia è associata ad un controller dedicato che gestisce gli eventi e la comunicazione con il modello.

Il documento è strutturato nelle seguenti sezioni:

- **Design Architetturale**: analisi della struttura del sistema e dei ruoli dei suoi componenti (Model, View, Controller).
- **Scenari di Casi d'Uso Ampliati con MVC**: flussi di interazione e scenari estesi, con una particolare attenzione agli scenari normali e alternativi.
- **Diagrammi di Sequenza e delle Classi**: rappresentazioni grafiche delle interazioni e della struttura del sistema.
- **Diagrammi delle Attività**: illustrazioni dei processi e delle dinamiche tra i componenti del sistema.
- **Design dell'Interfaccia Utente**: descrizione delle scelte progettuali relative all'interfaccia grafica e alla separazione tra layout e logica di business.

2 Design Architetturale

Per strutturare il sistema, abbiamo optato per un approccio MVC, un pattern in cui il sistema è separato in tre componenti principali:

- **Model**: Gestisce i dati e la logica di business.
- **View**: Rappresenta l'interfaccia utente.
- **Controller**: Gestisce gli input dell'utente e aggiorna il modello.

Questo approccio consente una chiara separazione delle responsabilità, facilita il riutilizzo del codice e rende il sistema più manutenibile e scalabile.

2.1 View

Per ogni Interfaccia GUI con cui l'utente interagirà è stata strutturato un file FXML che ne gestisce il layout, associando ad ognuno un file .css che ne gestisca il design. Un esempio pratico è la gestione dell'interfaccia per la rubrica dell'utente:
il file *InterfacciaUtente.fxml* definisce la struttura grafica, il file *style.css* ne personalizza il design, e il controller *InterfacciaUtenteController* si occupa della validazione dei dati inseriti e della comunicazione con il Model.

2.2 Controller

Ad ogni interfaccia Grafica è stato associato un Controller, che alloca e gestisce le strutture dati per una buona esperienza utente.

2.3 Model

Il Model non solo contiene i dati interni ma implementa anche la logica per validare e memorizzare le informazioni.

3 Scenari di Casi d'Uso Ampliati con MVC

Creazione Contatto

Campo	Descrizione
Attori partecipanti	Utente (Attore Principale), Sistema (Model, View, Controller)
Pre-condizioni	L'interfaccia grafica è stata correttamente caricata. Sono state memorizzate correttamente le informazioni precedenti.
Post-condizioni	Il contatto è salvato nel sistema ed è visibile nella rubrica.
Flusso normale	1. L'utente accede all'interfaccia utente (View) e preme il pulsante "Aggiungi Contatto". 2. Compila i campi richiesti. 3. La View raccoglie i dati e li invia al Controller tramite il pulsante "Salva". 4. Il Controller valida i dati e li passa al Model. 5. Il Model salva i dati e restituisce un messaggio di conferma al Controller. 6. Il Controller aggiorna la View con un messaggio di successo o errore.
Flussi alternativi	• L'utente lascia alcuni campi vuoti (la View mostra un messaggio di errore). • Si verifica un errore durante il salvataggio (esempio: database non disponibile).

Modifica Contatto

Campo	Descrizione
Attori partecipanti	Utente (Attore Principale), Sistema (Model, View, Controller).
Pre-condizioni	La lista dei contatti è stata caricata correttamente nella View. Il contatto selezionato esiste.
Post-condizioni	Il contatto viene aggiornato con i nuovi dati. La lista dei contatti viene aggiornata nella View per riflettere le modifiche.
Flusso normale	1. L'utente seleziona un contatto dalla lista mostrata nella schermata principale (View). 2. La View invia un evento al Controller per recuperare i dettagli del contatto selezionato. 3. Il Controller comunica con il Model per ottenere i dati del contatto. 4. Il Controller restituisce i dati alla View, che li visualizza in un modulo di modifica (file FXML specifico). 5. L'utente aggiorna i campi desiderati (es. nome, numero di telefono, email) e preme il pulsante di salvataggio. 6. La View invia i dati aggiornati al Controller. 7. Il Controller valida i dati modificati. 8. Se la validazione ha successo, il Controller invia i dati aggiornati al Model per essere salvati nel database. 9. Il Model aggiorna il record nel database e notifica il Controller dell'esito positivo. 10. Il Controller informa la View, che aggiorna la lista dei contatti e mostra un messaggio di successo all'utente.

Flussi alternativi	• L'utente chiude il modulo di modifica senza apportare cambiamenti. La View torna alla schermata principale senza notificare il Controller. • Se uno o più campi non sono validi (es. email non corretta), il Controller aggiorna la View con un messaggio di errore. • Se si verifica un problema durante l'aggiornamento del database, il Controller notifica l'errore alla View, che mostra un messaggio all'utente.

Eliminazione Contatto

Campo	Descrizione
Attori partecipanti	Utente (Attore Principale). Sistema (Model, View, Controller).
Pre-condizioni	La lista dei contatti è stata caricata correttamente nella View. Il contatto da eliminare esiste.
Post-condizioni	Il contatto viene rimosso dal database. La lista dei contatti viene aggiornata nella View, senza includere il contatto eliminato.
Flusso normale	1. L'utente seleziona un contatto dalla lista mostrata nella schermata principale (View). 2. L'utente preme il pulsante "Elimina". 3. La View invia un evento al Controller indicando il contatto da eliminare. 4. Il Controller comunica con il Model per rimuovere il contatto dal database. 5. Il Model elimina il record corrispondente dal database e restituisce l'esito al Controller. 6. Se l'eliminazione è andata a buon fine, il Controller notifica la View. 7. La View aggiorna la lista dei contatti rimuovendo il contatto eliminato e mostra un messaggio di successo all'utente.
Flussi alternativi	Se si verifica un problema durante l'eliminazione, il Controller informa la View, che mostra un messaggio di errore all'utente.

Salvataggio e Caricamento Contatti

Campo	Descrizione
Attori partecipanti	Utente (Attore Principale) Sistema (Model, View, Controller).
Pre-condizioni	Esiste almeno un contatto salvato.
Post-condizioni	L'elenco dei contatti è visualizzato all'utente.
Flusso normale	1. All'avvio dell'applicazione, la View della lista dei contatti ('Contacts.fxml') viene inizializzata. 2. La View invia una richiesta al Controller per recuperare i dati dei contatti. 3. Il Controller richiede al Model l'elenco completo dei contatti. 4. Il Model recupera i dati dal database o da una struttura dati persistente. 5. Il Model restituisce l'elenco dei contatti al Controller. 6. Il Controller invia i dati alla View. 7. La View popola la lista dei contatti e li mostra all'utente.
Flussi alternativi	• Il Model restituisce un elenco vuoto e la View mostra un messaggio del tipo "Nessun contatto disponibile". • Il Model genera un errore. Il Controller cattura l'errore e aggiorna la View per notificare all'utente che il caricamento dei contatti non è riuscito.

Ricerca Contatto

Campo	Descrizione
Attori partecipanti	Utente (Attore Principale) Sistema (Model, View, Controller).
Pre-condizioni	La lista dei contatti è stata caricata correttamente nella View. Il campo di ricerca è visibile e interattivo.
Post-condizioni	La View mostra i contatti che corrispondono al criterio di ricerca.
Flusso normale	1. L'utente digita un criterio di ricerca (ad esempio, nome o numero di telefono) nel campo di ricerca fornito dalla View. 2. La View invia il criterio di ricerca al Controller tramite un evento (ad esempio, clic su un pulsante "Cerca"). 3. Il Controller riceve il criterio di ricerca ed esegue la ricerca nel database o nella struttura dati. 4. Il Model filtra i contatti in base al criterio e restituisce un elenco di risultati al Controller. 5. Il Controller invia i risultati alla View. 6. La View aggiorna la lista visualizzata mostrando solo i contatti corrispondenti al criterio di ricerca.
Flusso alternativo	Il Model restituisce un elenco vuoto. La View mostra un messaggio del tipo "Nessun contatto trovato".

Gestione del *Menù Preferiti*

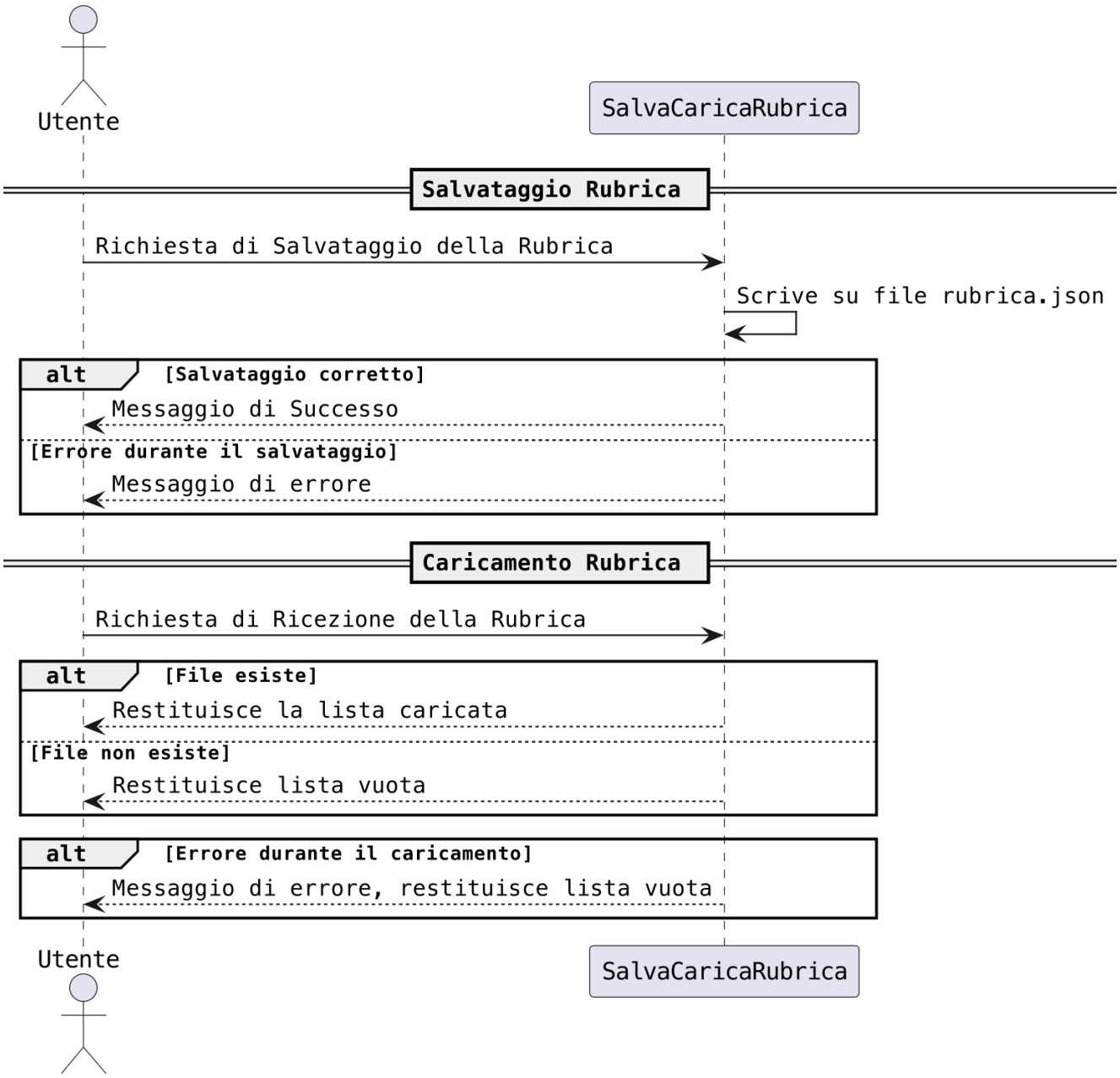
Campo	Descrizione
Attori partecipanti	Utente, Sistema
Pre-condizioni	La lista dei preferiti (`preferitiList`) è correttamente inizializzata. La rubrica completa (`contactList`) è disponibile e non vuota.
Post-condizioni	Il contatto selezionato viene aggiunto alla lista dei preferiti. La `ListView` dei preferiti è aggiornata dinamicamente.
Flusso normale	1. **Visualizzazione del Menu dei Preferiti:** • L'utente accede al menu dei preferiti. • Il sistema carica la lista dei preferiti esistenti utilizzando l'oggetto `ObservableList<Contatto>`. • La lista è visualizzata nella `ListView`, con ogni contatto formattato tramite il metodo `creaLabelContatto()`. 2. **Aggiunta di un Contatto ai Preferiti**: • L'utente preme il pulsante "Aggiungi ai Preferiti". • Il sistema apre una nuova finestra (`SelezionaContattiDaRubrica.fxml`) che mostra la rubrica completa. • L'utente cerca il contatto desiderato utilizzando la barra di ricerca dinamica. • Dopo aver selezionato un contatto, l'utente preme "Aggiungi". • Il contatto viene aggiunto alla lista dei preferiti se non è già presente. • La finestra di selezione si chiude, e l'utente ritorna al menu dei preferiti. 3. **Modifica dei Preferiti** (attualmente non implementato): • L'utente preme il pulsante "Modifica". • (Scenario ipotetico) Il sistema consente di aggiornare o rimuovere contatti dalla lista dei preferiti.
Flussi alternativi	• Il sistema rileva la duplicazione e mostra un messaggio informativo ("Il contatto è già nei preferiti"). • Nessuna azione viene intrapresa e il sistema ritorna al menu dei preferiti senza modifiche.

Visualizzazione Contatto

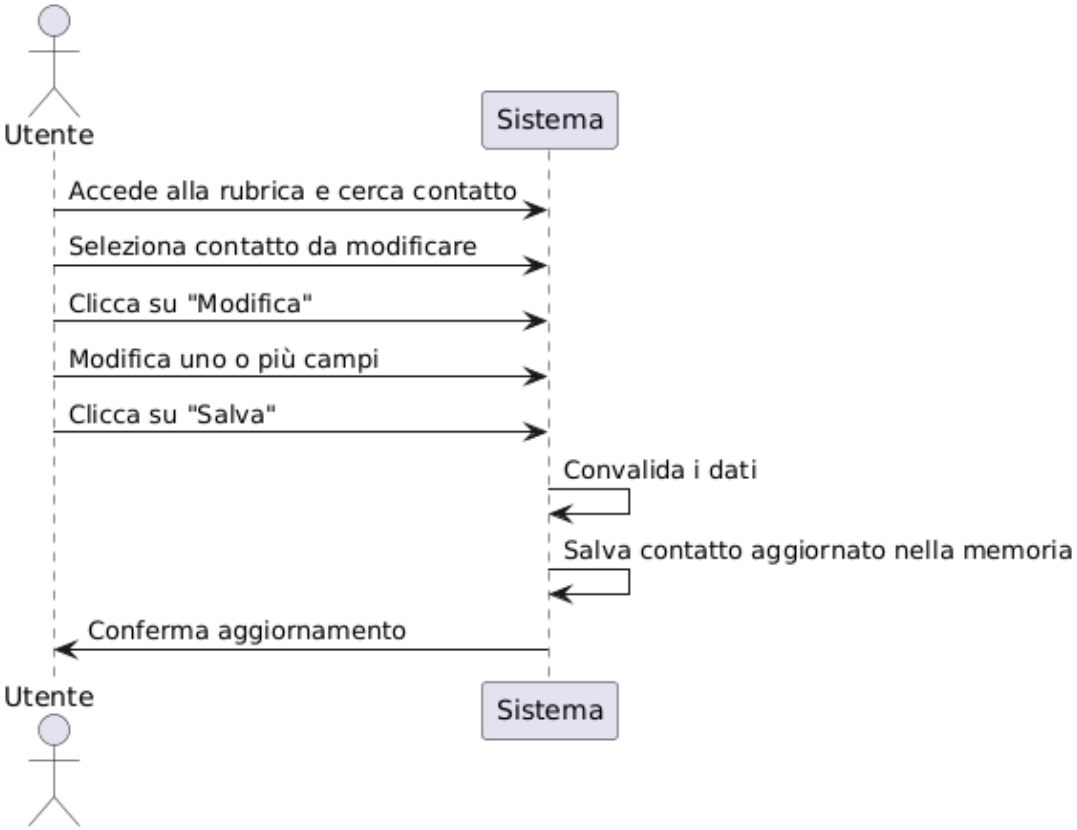
Campo	Descrizione
Attori partecipanti	Utente (Attore Principale) Sistema (Model, View, Controller).
Pre-condizioni	La lista dei contatti è caricata correttamente e visualizzata nella parte sinistra dello SplitPane.
Post-condizioni	I dettagli del contatto selezionato sono visualizzati nella parte destra dello SplitPane.
Flusso normale	1. L'utente seleziona un contatto dalla lista della rubrica visualizzata nella parte sinistra dello SplitPane. 2. Il sistema cattura l'evento di selezione tramite un listener sulla `ListView`. 3. Il Controller recupera i dettagli del contatto selezionato. 4. La View aggiorna la parte destra dello SplitPane mostrando le informazioni dettagliate del contatto.
Flussi alternativi	• La parte destra dello SplitPane rimane vuota o mostra un messaggio predefinito ("Seleziona un contatto per visualizzarne i dettagli"). • La `ListView` rimane vuota e la parte destra dello SplitPane mostra un messaggio ("Nessun contatto disponibile").

4 Diagrammi di sequenza

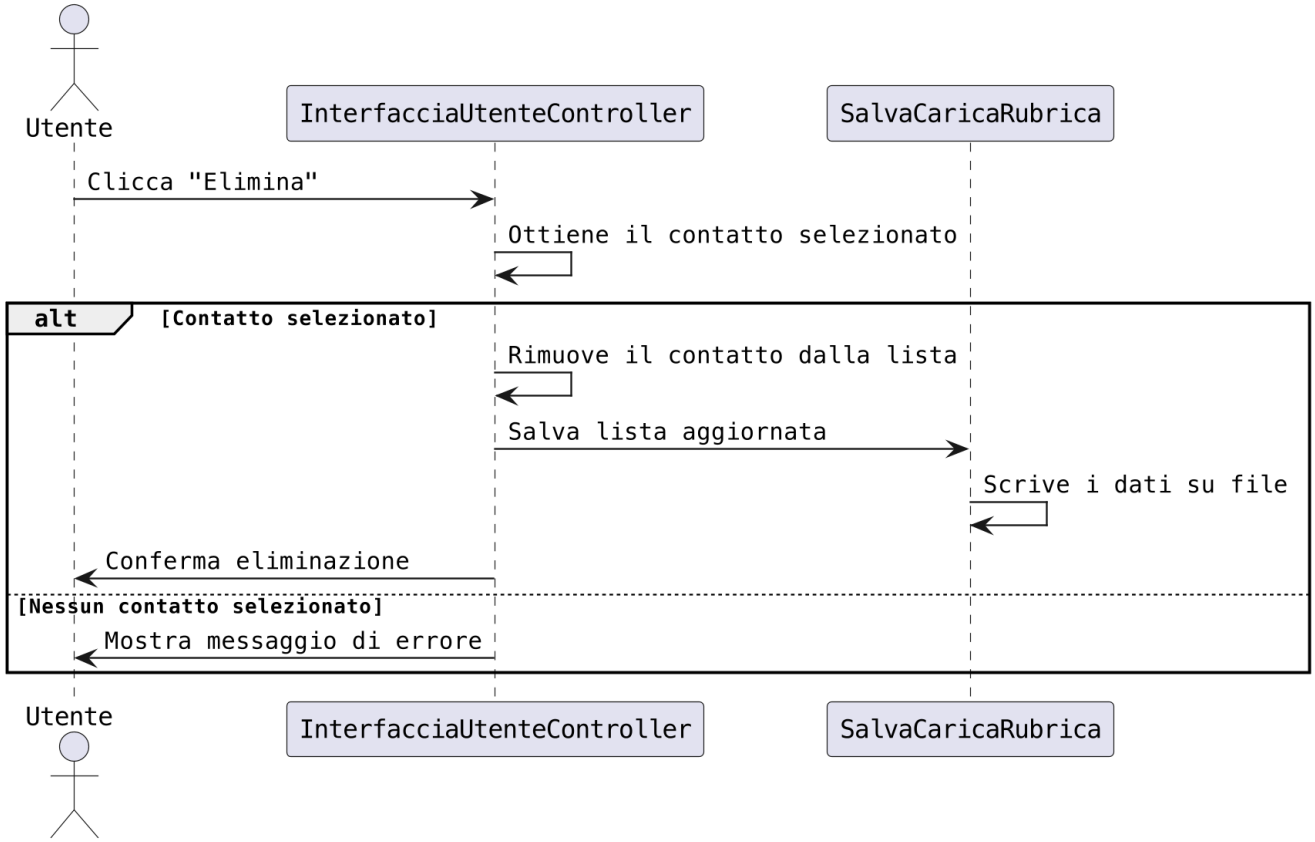
Creazione Contatto



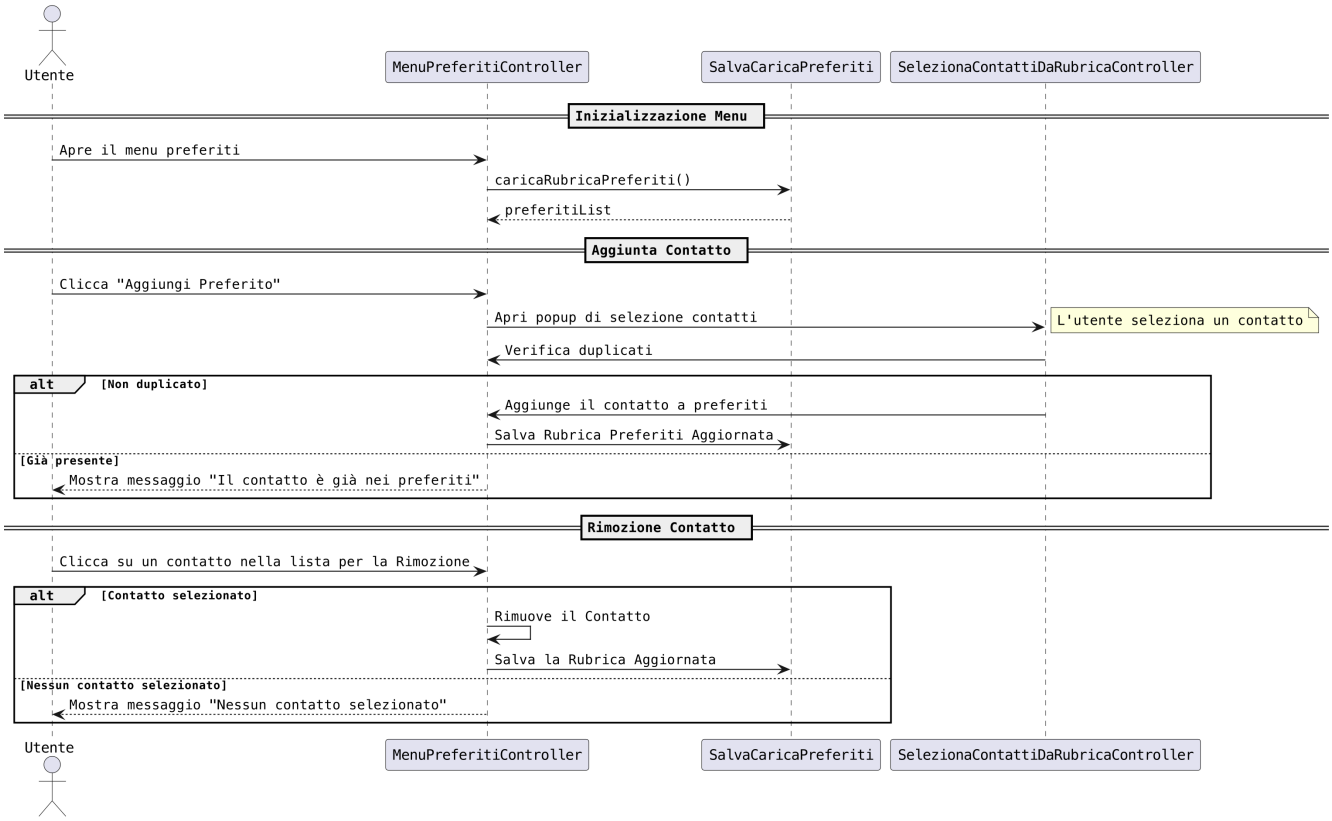
Modifica Contatto



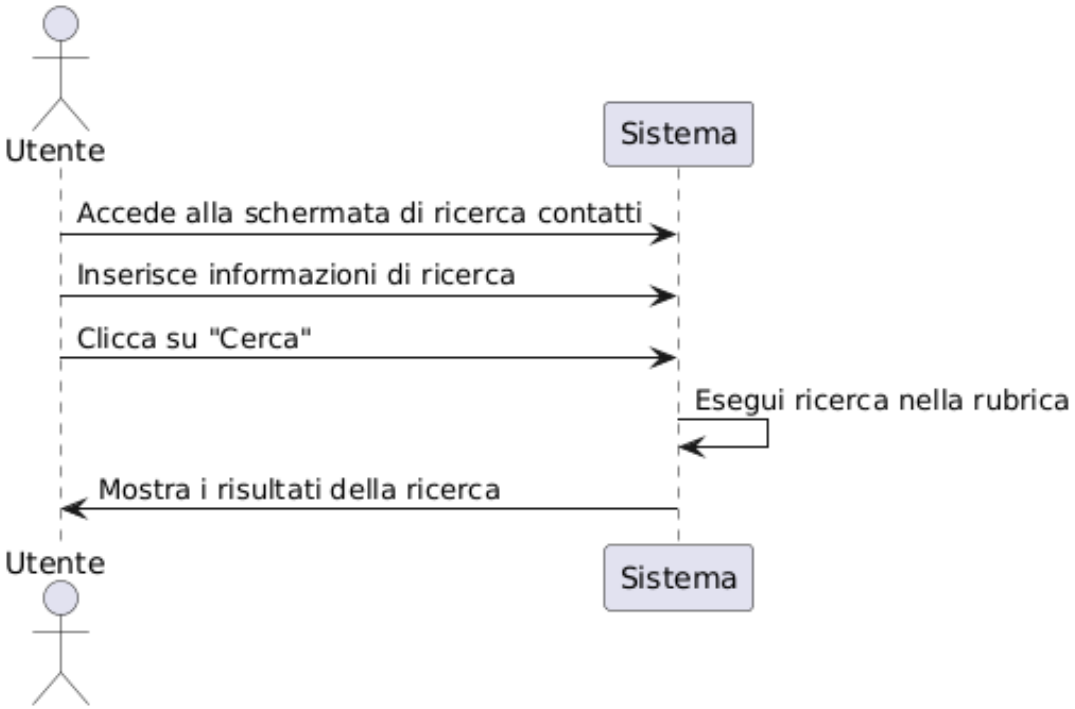
Eliminazione Contatto



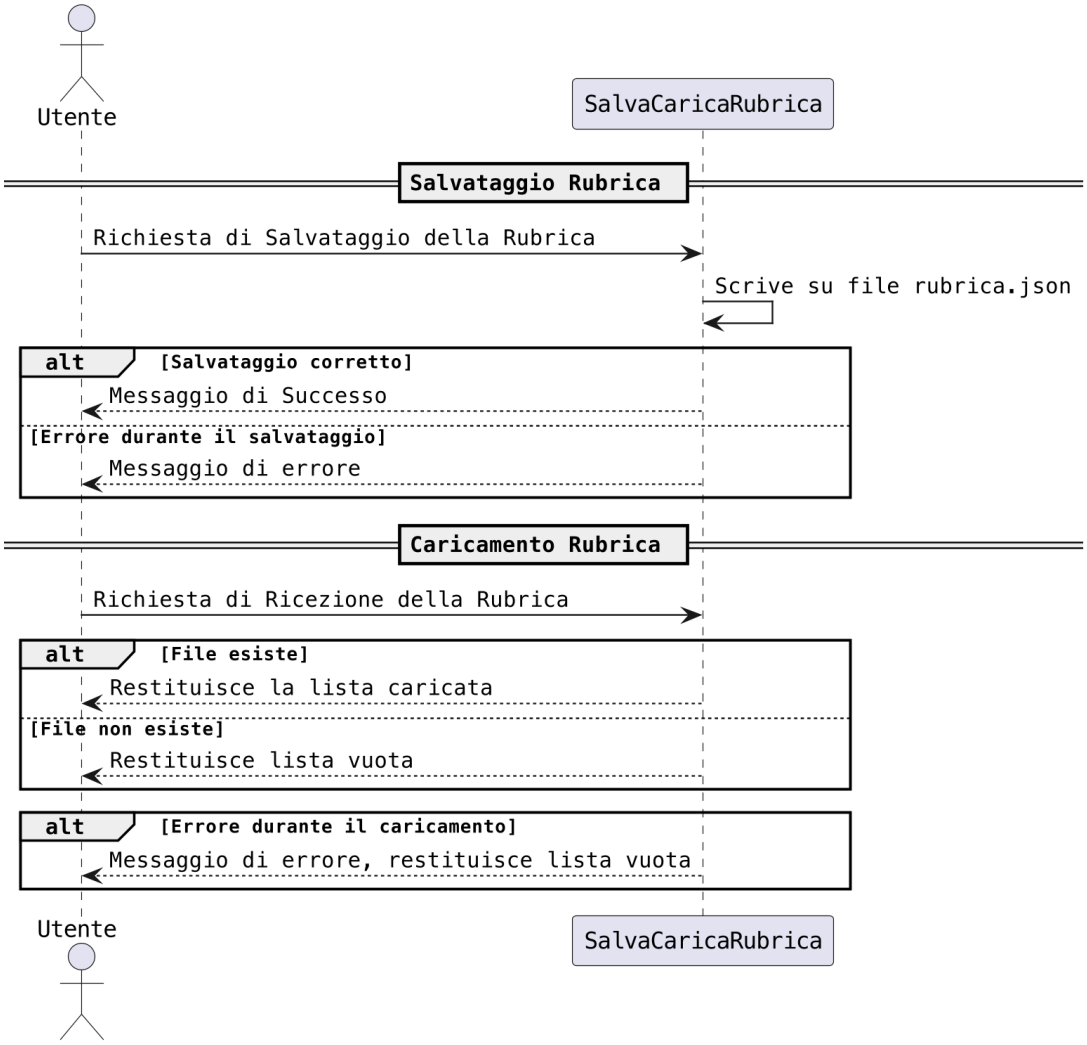
Gestione del Menu Preferiti



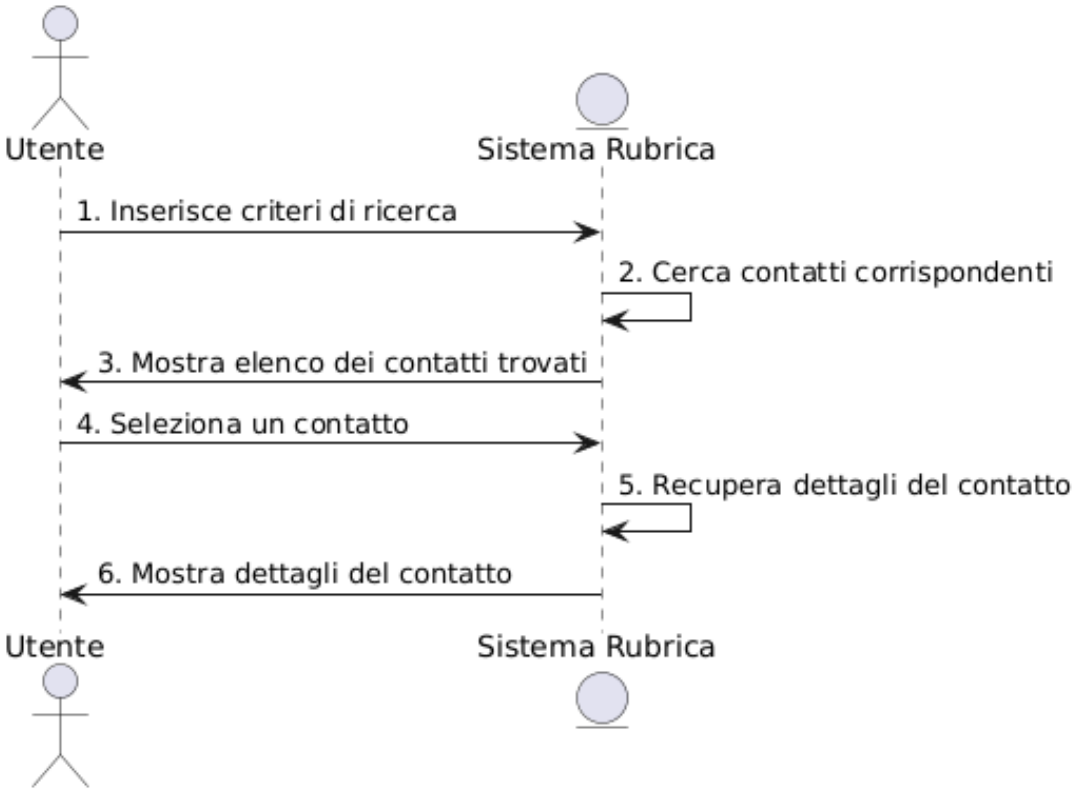
Ricerca Contatto



Salvataggio e Caricamento Contatti



Visualizzazione Contatti



5 Diagramma delle Classi

Il diagramma delle classi mostrato nella figura sottostante illustra le principali classi e le loro relazioni. Le classi sono collegate tramite associazioni e dipendenze, che rappresentano il flusso di informazioni e le interazioni tra i vari componenti dell'applicazione.

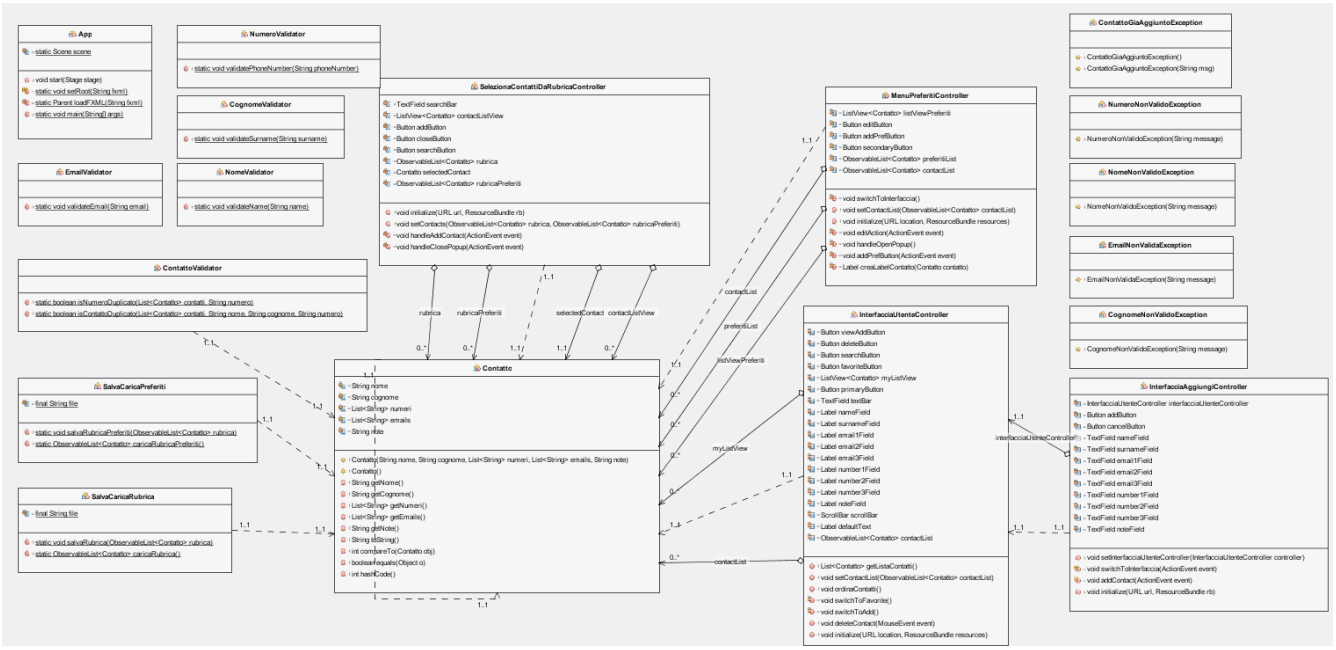


Figura 1: DiagrammadelleClassi

- **Contatto**: Gestisce i dati di un contatto (nome, cognome, numeri di telefono, email e note), consentendo operazioni come aggiungere, modificare, visualizzare e ordinare contatti. Supporta la serializzazione e deserializzazione tramite JSON e implementa l'interfaccia `Comparable` per l'ordinamento case-insensitive.
- **InterfacciaAggiungiController**: Gestisce l'interfaccia per l'aggiunta di contatti, raccogliendo e validando i dati inseriti (nome, cognome, numeri di telefono, email). Aggiunge il contatto, lo ordina e salva la rubrica su file, gestendo anche i duplicati.
- **InterfacciaUtenteController**: Gestisce l'interfaccia utente per visualizzare e aggiornare la lista dei contatti, e permette operazioni come aggiungere, cercare e rimuovere contatti. Implementa l'interfaccia `Initializable` per caricare i contatti da file e usa JavaFX per la gestione dell'interfaccia.
- **MenuPreferitiController**: Gestisce la visualizzazione e gestione dei contatti preferiti, interagendo con altre classi come `Contatto` e `SalvaCaricaPreferiti`, e gestendo eccezioni come `ContattoGiaAggiuntoException`.
- **SelezionaContattiDaRubricaController**: Permette di selezionare contatti dalla rubrica e aggiungerli ai preferiti, con una barra di ricerca dinamica. Gestisce l'aggiunta dei contatti ai preferiti e salva automaticamente la lista aggiornata.
- **SalvaCaricaPreferiti**: Gestisce il salvataggio e caricamento dei contatti preferiti in un file JSON, utilizzando la libreria Jackson per la serializzazione e deserializzazione.

- **SalvaCaricaRubrica**: Gestisce il salvataggio e caricamento della rubrica in formato JSON, interagendo con la libreria Jackson e utilizzando `ObservableList` per mantenere la lista aggiornata in tempo reale.
- **ContattoValidator**: Contiene metodi per validare la duplicazione di un contatto, sia per numero di telefono che per dati completi.
- **EmailValidator**: Valida un indirizzo email, verificando che segua un formato valido.
- **NomeValidator e CognomeValidator**: Validano rispettivamente il nome e il cognome di un contatto.
- **Classi di eccezione**: Le classi `CognomeNonValidoException`, `NomeNonValidoException`, `EmailNonValidaException` e `NumeroNonValidoException` rappresentano eccezioni per la gestione di dati non validi, estendendo la classe `Exception` e fornendo messaggi di errore personalizzati.

6 Diagramma delle Attività

In questo diagramma delle attività si vuole dare al cliente una panoramica generale delle funzionalità del sistema. Nota: alcune specificità sono state omesse per una visualizzazione più semplice e intuitiva.

In particolare:

- L'utente avvia l'applicazione, che carica automaticamente la rubrica salvata in precedenza. Se è il primo accesso, la rubrica sarà vuota.
- L'utente può effettuare una delle seguenti azioni:
  1. **Aggiungere un nuovo contatto**:
     - L'utente inserisce i dettagli del contatto (nome, cognome, numero di telefono, email, note).
     - Il contatto viene salvato e aggiunto alla rubrica.
  2. **Modificare un contatto esistente**:
     - L'utente seleziona un contatto dalla rubrica.
     - Effettua le modifiche desiderate (nome, cognome, numeri, email o note).
     - Le modifiche vengono salvate.
  3. **Eliminare un contatto esistente**:
     - L'utente seleziona il contatto da eliminare.
     - Il sistema chiede una conferma prima di procedere.
     - Se confermato, il contatto viene eliminato e la rubrica aggiornata.
  4. **Cercare un contatto**:
     - L'utente inserisce i criteri di ricerca (es. nome o numero di telefono).
     - Il sistema mostra i risultati corrispondenti.
  5. **Visualizzare i dettagli di un contatto**: L'utente seleziona un contatto e il sistema ne mostra i dettagli (nome, cognome, numeri, email e note).
- **Conclusione dell'azione**: Dopo aver completato un'operazione, l'utente può:
  - Uscire dall'applicazione.
  - Tornare alla schermata principale per selezionare un'altra azione.

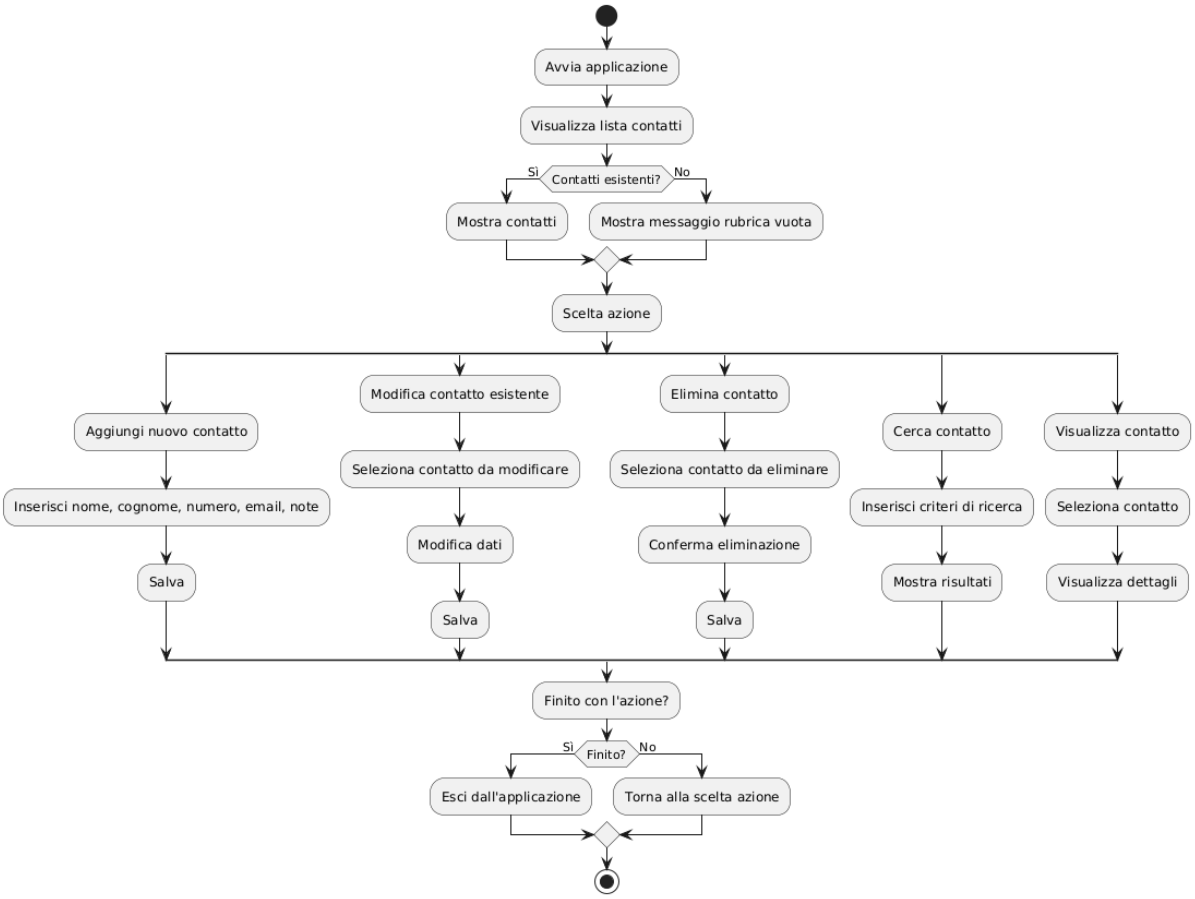


Figura 2: Diagramma delle Attività

7 Design dell' Interfaccia Utente

Il design dell'interfaccia utente è stato sviluppato sulla base dei casi d'uso identificati, garantendo una navigazione chiara e fluida tra le diverse schermate dell'applicazione. L'obiettivo è creare un'esperienza utente semplice e intuitiva, che permetta di gestire la rubrica in modo efficace. Di seguito vengono descritti i dettagli di ciascuna interfaccia e le modalità di navigazione tra di esse:

- **Schermata principale** (*InterfacciaUtenteController*)
  La schermata principale mostra una lista di contatti. L'utente può aggiungere nuovi contatti, modificare o eliminare quelli esistenti. Da questa schermata è possibile:
  - **Aggiungere un nuovo contatto**: L'utente può cliccare su un pulsante per accedere alla schermata di aggiunta di un contatto: *InterfacciaAggiungiController*.
  - **Accedere ai preferiti**: Un altro pulsante consente di visualizzare i contatti preferiti tramite la schermata *MenuPreferitiController*.
- **Schermata *Aggiunta Contatto*** (*InterfacciaAggiungiController*)
  Questa schermata permette di inserire nuovi contatti nella rubrica. Include campi per il nome, il cognome, il numero di telefono, l'email e le note. Da questa schermata è possibile:
  - **Confermare e Salvare la creazione di un contatto**: Dopo aver completato il modulo, l'utente può cliccare su un pulsante per aggiungere il contatto alla rubrica. L'operazione aggiorna la lista principale dei contatti e ritorna alla schermata principale.
  - **Annullare la creazione di un contatto**: Se l'utente non vuole aggiungere il contatto, può annullare l'operazione e tornare alla schermata principale senza modifiche.
- **Schermata *Menù Preferiti*** (*MenuPreferitiController*)
  La schermata dei preferiti mostra i contatti contrassegnati come preferiti. Gli utenti possono visualizzare, aggiungere o rimuovere contatti dai preferiti. Da questa schermata è possibile:
  - **Selezionare contatti dalla rubrica**: L'utente può cliccare su un pulsante che apre una finestra pop-up *SelezionaContattiDaRubricaController* per selezionare i contatti da aggiungere ai preferiti.
  - **Tornare alla schermata principale**: Un altro pulsante consente di tornare alla schermata principale senza modificare i preferiti.

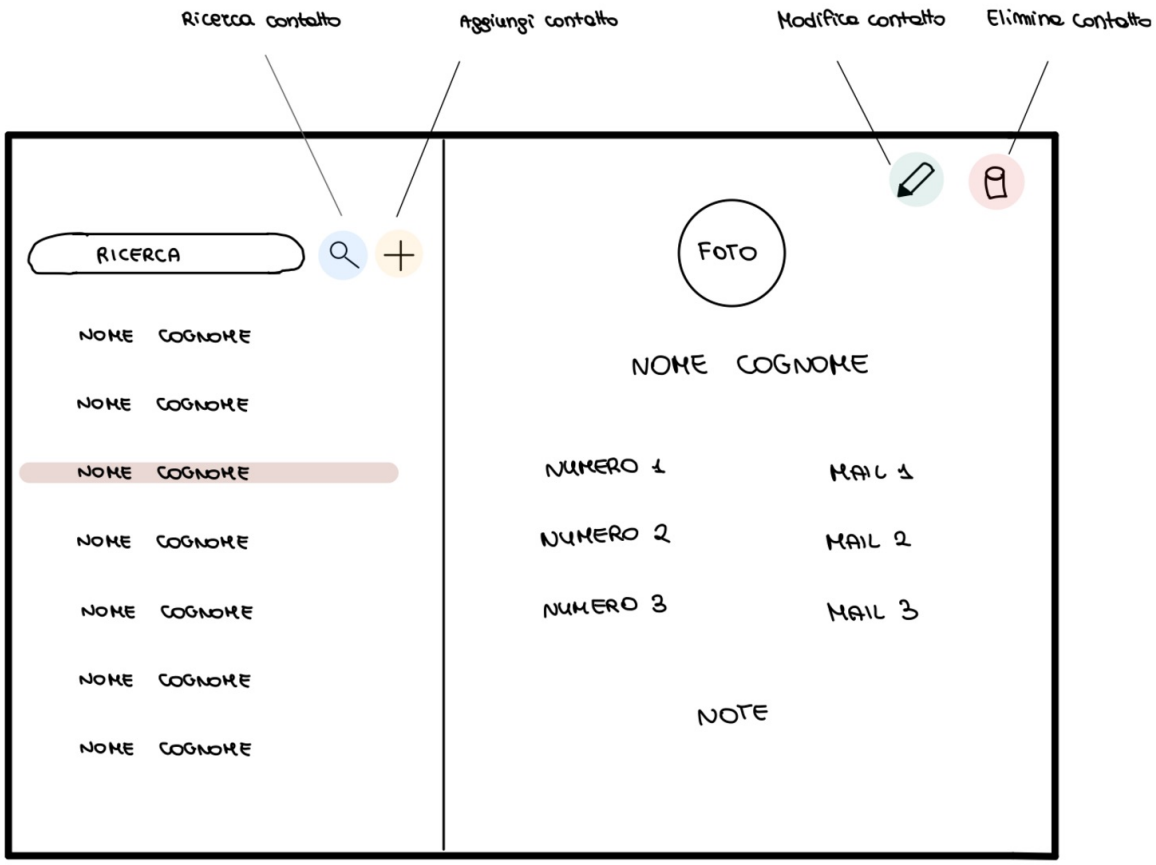


Figura 3: schizzo